**World Subclasses**

**Level\_One**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class Level\_One here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Level\_One extends World

{

/\*\*

\* Constructor for objects of class Level\_One.

\*

\*/

public static final int WIDE = 1000;

public static final int HIGH = 800;

public static final int CELL = 1;

GreenfootSound backgroundMusic = new GreenfootSound("stage1.wav");

/\*\*

\* For some reason, the background music wouldn't play the same way

\* as the sound effects that play when the blob picks up items or

\* jumps. Only way to get it to play was to declare it like so

\*/

SimpleTimer tim = new SimpleTimer(); //Creates a timer

Counter CountDown = new Counter(); //Creates the yellow counter

CounterP JewelCount = new CounterP(); //Creates the purple counter

int start = 1;

int stagestart = 1;

//These two int variables help out with starting the timer and

//Playing the background music

Ghost ghost = new Ghost();

public Level\_One()

{

super(WIDE, HIGH, CELL, false);

addObject(ghost,75,760);

addObject(CountDown,80,50);

CountDown.setValue(60);

getBackground().drawImage(new GreenfootImage("TIME", 30, Color.WHITE, null), 50, 10);

addObject(JewelCount,180,50);

JewelCount.setValue(0);

getBackground().drawImage(new GreenfootImage("JEWELS", 30, Color.WHITE, null), 140, 10);

//Declares the values of the counters

CreatePlatforms();

PlaceJewels();

PlaceClocks();

PlaceEnemies();

PlacePotions();

}

public void PlacePortal()

{

PortalToNextLvl portal = new PortalToNextLvl();

addObject(portal,35,120);

}

public void PlaceClocks()

{

Clock clock = new Clock();

addObject(clock,325,750);

Clock clock2 = new Clock();

addObject(clock2,900,215);

}

public void PlaceEnemies()

{

Enemy enemy = new Enemy(425,525,1,0,0);

addObject(enemy,475,580);

}

public void TimeDisplay()

{

TIMEDISPLAY time = new TIMEDISPLAY();

addObject(time,80,20);

}

public void PlacePotions()

{

Potion potion = new Potion();

addObject(potion,35,120);

}

public void PlaceJewels()

{

/\*\*

\*This function creates jewels on preset coordinates

\*/

Jewel jewel = new Jewel();

addObject(jewel,200,720);

Jewel jewel2 = new Jewel();

addObject(jewel2,325,620);

Jewel jewel3 = new Jewel();

addObject(jewel3,50,400);

Jewel jewel4 = new Jewel();

addObject(jewel4,900,750);

Jewel jewel5 = new Jewel();

addObject(jewel5,650,150);

}

public void CreatePlatforms()

{

/\*\*

\*This function creates platfroms on preset coordinates

\*/

BetterPlatform firstBox = new BetterPlatform(100,50,0,0,0,0,0);

addObject(firstBox,200,775);

BetterPlatform firstPlat = new BetterPlatform(150,30,0,0,0,0,0);

addObject(firstPlat,325,675);

BetterPlatform bigBox = new BetterPlatform(150,150,0,0,0,0,0);

addObject(bigBox,475,675);

BetterPlatform secondPlat = new BetterPlatform(150,30,0,0,0,0,0);

addObject(secondPlat,675,650);

BetterPlatform elevator = new BetterPlatform(200,30,0,0,2,250,620);

addObject(elevator,900,615);

BetterPlatform firstHighPlat = new BetterPlatform(300,30,0,0,0,0,0);

addObject(firstHighPlat,600,300);

BetterPlatform firstMovingWall = new BetterPlatform(30,100,465,700,1,0,0);

addObject(firstMovingWall,650,235);

BetterPlatform firstHighMovingPlat = new BetterPlatform(100,30,200,375,1,0,0);

addObject(firstHighMovingPlat,350,250);

BetterPlatform portalPlat = new BetterPlatform(150,200,0,0,0,0,0);

addObject(portalPlat,50,250);

BetterPlatform jumpPotPlat = new BetterPlatform(100,30,0,0,0,0,0);

addObject(jumpPotPlat,50,450);

}

public void act()

{

if (start == 1)

{

if(JewelCount.getValue() == 5) PlacePortal();

if (tim.millisElapsed() > 1000)

{

CountDown.add(-1);

tim.mark();

}

/\*\*

\* Greenfoot seems to only count in milliseconds rather than

\* seconds. With this, every time it passes 1000 milliseconds

\* (1 second) it will add -1 to the value of 60 which will

\* make it count down

\*/

if (CountDown.getValue() == 0)

{

ghost.DeathScreenLevelOne();

}

/\*\*

\* This constantly checks the value of the timer. As soon as it

\* hits 0 however, the game will go to the game over screen

\*/

if (stagestart == 1)

{

backgroundMusic.playLoop();

}

/\*\*

\* This starts the music. The reason why this is under the act

\* method and not under the world constructor is because placing

\* it there would make the music play constantly even if we

\* didn't click start. Whilst it is playing music as intended

\* This would've been incredibly annoying to us as developers

\* if we're having to listen to it constantly whilst working on

\* the game

\*/

}

}

public void IncrementJewel()

{

JewelCount.add(1);

}

public void IncrementClock()

{

CountDown.add(10);

}

//Allows for other classes to control the music and stop it when changing levels

public void SetStageStart(int newStageStart)

{

stagestart = newStageStart;

backgroundMusic.stop();

}

}

**Level\_Two**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class NewLvl here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Level\_Two extends World

{

public static final int WIDE = 1000;

public static final int HIGH = 800;

public static final int CELL = 1;

GreenfootSound backgroundMusic = new GreenfootSound("stage1.wav");

/\*\*

\* For some reason, the background music wouldn't play the same way

\* as the sound effects that play when the blob picks up items or

\* jumps. Only way to get it to play was to declare it like so

\*/

SimpleTimer tim = new SimpleTimer(); //Creates a timer

Counter CountDown = new Counter(); //Creates the yellow counter

CounterP JewelCount = new CounterP(); //Creates the purple counter

int start = 1;

int stagestart = 1;

//These two int variables help out with starting the timer and

//Playing the background music

Ghost ghost = new Ghost();

public Level\_Two()

{

super(WIDE, HIGH, CELL, false);

addObject(ghost,75,760);

addObject(CountDown,80,50);

CountDown.setValue(60);

getBackground().drawImage(new GreenfootImage("TIME", 30, Color.WHITE, null), 50, 10);

addObject(JewelCount,180,50);

JewelCount.setValue(0);

getBackground().drawImage(new GreenfootImage("JEWELS", 30, Color.WHITE, null), 140, 10);

PlacePlatforms();

PlaceJewels();

PlaceClocks();

PlaceEnemies();

Sword sword = new Sword();

addObject(sword, 850, 300);

}

public void PlaceClocks()

{

Clock clock = new Clock();

addObject(clock,275,475);

Clock clock2 = new Clock();

addObject(clock2,900,400);

}

public void PlacePortal()

{

PortalToNextLvl portal = new PortalToNextLvl();

addObject(portal,900,775);

}

public void PlaceEnemies()

{

Enemy enemy = new Enemy(425,625,3,0,0);

addObject(enemy,475,780);

Enemy enemy2 = new Enemy();

addObject(enemy2, 950, 315);

Enemy enemy3 = new Enemy(400,600,1,0,0);

addObject(enemy3, 450, 215);

}

public void PlaceJewels()

{

Jewel jewel = new Jewel();

addObject(jewel,200,650);

Jewel jewel2 = new Jewel();

addObject(jewel2,365,550);

Jewel jewel3 = new Jewel();

addObject(jewel3,200,275);

Jewel jewel4 = new Jewel();

addObject(jewel4,450,700);

Jewel jewel5 = new Jewel();

addObject(jewel5,650,450);

}

public void PlacePlatforms()

{

BetterPlatform firstBox = new BetterPlatform(100,30,0,0,0,0,0);

addObject(firstBox,175,725);

BetterPlatform firstPlat = new BetterPlatform(150,30,0,0,0,0,0);

addObject(firstPlat,75,625);

BetterPlatform jumpPotPlat = new BetterPlatform(100,30,0,0,0,0,0);

addObject(jumpPotPlat,175,525);

BetterPlatform bigBox = new BetterPlatform(100,290,0,0,0,0,0);

addObject(bigBox,275,655);

BetterPlatform secondPlat = new BetterPlatform(600,30,0,0,0,0,0);

addObject(secondPlat,700,525);

BetterPlatform fourthPlat = new BetterPlatform(100,30,0,0,0,0,0);

addObject(fourthPlat,50,425);

BetterPlatform firstHighPlat = new BetterPlatform(300,30,0,0,0,0,0);

addObject(firstHighPlat,500,250);

BetterPlatform secondHighPlat = new BetterPlatform(150,30,0,0,0,0,0);

addObject(secondHighPlat,200,325);

BetterPlatform thirdHighPlat = new BetterPlatform(300,30,0,0,0,0,0);

addObject(thirdHighPlat,850,350);

}

public void act()

{

if (start == 1)

{

if(JewelCount.getValue() == 5) PlacePortal();

if (tim.millisElapsed() > 1000)

{

CountDown.add(-1);

tim.mark();

}

/\*\*

\* Greenfoot seems to only count in milliseconds rather than

\* seconds. With this, every time it passes 1000 milliseconds

\* (1 second) it will add -1 to the value of 60 which will

\* make it count down

\*/

if (CountDown.getValue() == 0)

{

ghost.DeathScreenLevelTwo();

}

/\*\*

\* This constantly checks the value of the timer. As soon as it

\* hits 0 however, the game will go to the game over screen

\*/

if (stagestart == 1)

{

backgroundMusic.playLoop();

}

/\*\*

\* This starts the music. The reason why this is under the act

\* method and not under the world constructor is because placing

\* it there would make the music play constantly even if we

\* didn't click start. Whilst it is playing music as intended

\* This would've been incredibly annoying to us as developers

\* if we're having to listen to it constantly whilst working on

\* the game

\*/

}

}

public void IncrementJewel()

{

JewelCount.add(1);

}

public void IncrementClock()

{

CountDown.add(10);

}

//Allows for other classes to control the music and stop it when changing levels

public void SetStageStart(int newStageStart)

{

stagestart = newStageStart;

backgroundMusic.stop();

}

}

**WinLevel**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class Level\_Two here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class WinLevel extends World

{

public static final int WIDE = 1000;

public static final int HIGH = 800;

public static final int CELL = 1;

GreenfootSound backgroundMusic = new GreenfootSound("stage2.mp3");

int stagestart = 1;

Ghost ghost = new Ghost();

public WinLevel()

{

super(WIDE, HIGH, CELL, false);

addObject(ghost,75,760);

CreatePlatforms();

CreateCookie();

//Declares the values of the counters

}

public void SetStageStart(int newStageStart)

{

stagestart = newStageStart;

backgroundMusic.stop();

}

public void CreateCookie()

{

Cookie cookie = new Cookie();;

addObject(cookie,850, 475);

}

public void CreatePlatforms()

{

/\*\*

\*This function creates platfroms on preset coordinates

\*/

BetterPlatform box1 = new BetterPlatform(800,50,0,0,0,0,0);

addObject(box1,600,775);

BetterPlatform box2 = new BetterPlatform(800,50,0,0,0,0,0);

addObject(box2,650,725);

BetterPlatform box3 = new BetterPlatform(800,50,0,0,0,0,0);

addObject(box3,700,675);

BetterPlatform box4 = new BetterPlatform(800,50,0,0,0,0,0);

addObject(box4,750,625);

BetterPlatform box5 = new BetterPlatform(800,50,0,0,0,0,0);

addObject(box5,800,575);

BetterPlatform box6 = new BetterPlatform(800,50,0,0,0,0,0);

addObject(box6,850,525);

}

public void act()

{

if (stagestart == 1)

{

backgroundMusic.playLoop();

}

}

}

**title**

**import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)**

**/\*\***

**\* Write a description of class YouWonScreen here.**

**\***

**\* @author (your name)**

**\* @version (a version number or a date)**

**\*/**

**public class title extends World**

**{**

**/\*\***

**\* Constructor for objects of class YouWonScreen.**

**\***

**\*/**

**//GreenfootSound titleMusic = new GreenfootSound("title.mp3");**

**int stagestart = 1;**

**public title()**

**{**

**// Create a new world with 600x400 cells with a cell size of 1x1 pixels.**

**super(1000, 800, 1);**

**}**

**public void act()**

**{**

**if (Greenfoot.mousePressed(this))**

**{**

**Greenfoot.setWorld(new Level\_One());**

**}**

**}**

**}**

**GameOverScreen**

**import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)**

**/\*\***

**\* Write a description of class GameOverScreen here.**

**\***

**\* @author (your name)**

**\* @version (a version number or a date)**

**\*/**

**public class GameOverScreen extends World**

**{**

**/\*\***

**\* Constructor for objects of class GameOverScreen.**

**\***

**\*/**

**//Initalises global variables**

**GreenfootSound backgroundMusic = new GreenfootSound("youlose.mp3");**

**int stagestart = 1;**

**PlayAgain button = new PlayAgain();**

**public GameOverScreen()**

**{**

**// Create a new world with 1000x800 cells with a cell size of 1x1 pixels.**

**super(1000, 800, 1);**

**addObject(button,500,340);**

**}**

**public void act()**

**{**

**//If button pressed then moves the user to Level One starting the game over**

**if (Greenfoot.mousePressed(button))**

**{**

**Greenfoot.setWorld(new Level\_One());**

**}**

**//Starts the music**

**if (stagestart == 1);**

**{**

**backgroundMusic.playLoop();**

**}**

**}**

**}**

**YouWonScreen**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class YouWonScreen here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class YouWonScreen extends World

{

/\*\*

\* Constructor for objects of class YouWonScreen.

\*

\*/

GreenfootSound backgroundMusic = new GreenfootSound("youwin.mp3");

int stagestart = 1;

public YouWonScreen()

{

// Create a new world with 600x400 cells with a cell size of 1x1 pixels.

super(1000, 800, 1);

PlayAgain button = new PlayAgain();

addObject(button,490,390);

}

public void act()

{

//If button pressed then moves the user to Level One starting the game over

if (Greenfoot.mousePressed(this))

{

Greenfoot.setWorld(new Level\_One());

}

if (stagestart == 1);

{

backgroundMusic.playLoop();

}

}

}

**Actor Classes**

**BetterPlatform**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class BetterPlatform here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class BetterPlatform extends Actor

{

private int speed;

private int leftTurn;

private int rightTurn;

private int up;

private int down;

public BetterPlatform(){

this(100, 25,0,0,0,0,0);

}

public BetterPlatform(

int width,

int height,

int leftTurnChanged,

int rightTurnChanged,

int speedChanged,

int upChanged,

int downChanged)

{

GreenfootImage image = getImage();

image.scale(width,height);

setImage(image);

setLeftTurn(leftTurnChanged);

setRightTurn(rightTurnChanged);

setSpeed(speedChanged);

setUp(upChanged);

setDown(downChanged);

}

public void act()

{

//controls the platform if it is meant to be moving left and right

if(leftTurn != 0 || rightTurn != 0)

{

setLocation(getX() + speed,getY());

Actor actor = getOneIntersectingObject(null);

if(actor!=null)

{

actor.setLocation(actor.getX()+speed,actor.getY());

}

if(atTurningPointX())

{

speed = -speed;

}

}

//controls the platform if it is meant to be moving up and down

if(up != 0 || down != 0)

{

setLocation(getX(),getY()+ speed);

Actor actor = getOneIntersectingObject(null);

if(actor!=null)

{

actor.setLocation(actor.getX(),actor.getY()+speed);

}

if(atTurningPointY())

{

speed = -speed;

}

}

}

public void setLeftTurn(int leftTurnChanged)

{

leftTurn = leftTurnChanged;

}

public void setRightTurn(int rightTurnChanged)

{

rightTurn = rightTurnChanged;

}

//Allows for speed to be changed by other classes

public void setSpeed(int speedChanged)

{

speed = speedChanged;

}

//Allows for up to be changed by other classes

public void setUp(int upChanged)

{

up = upChanged;

}

//Allows for down attribute to be changed by other classes

public void setDown(int downChanged)

{

down = downChanged;

}

public boolean atTurningPointX()

{

return (getX() <= leftTurn || getX() >= rightTurn);

}

public boolean atTurningPointY()

{

return (getY() <= up || getY() >= down);

}

}

**Clock**

**import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)**

**/\*\***

**\* Write a description of class Clock here.**

**\***

**\* @author (your name)**

**\* @version (a version number or a date)**

**\*/**

**public class Clock extends Actor**

**{**

**/\*\***

**\* Act - do whatever the Clock wants to do. This method is called whenever**

**\* the 'Act' or 'Run' button gets pressed in the environment.**

**\*/**

**int counter = 0;**

**int interval = 2;**

**public Clock()**

**{**

**GreenfootImage image = getImage();**

**image.scale(image.getWidth()/2, image.getHeight()/2);**

**}**

**public void act()**

**{**

**//Allows for the object to hover in place.**

**//Every 10 frames the objects moves and when it reachs a top point it stops**

**//and changes it's direction**

**if(counter == 0)**

**{**

**setLocation(getX(),getY() - interval);**

**counter = counter + 1;**

**}**

**else if(counter == 10)**

**{**

**setLocation(getX(),getY() - interval);**

**counter = counter + 1;**

**}**

**else if(counter == 20)**

**{**

**setLocation(getX(),getY() - interval);**

**counter = counter + 1;**

**}**

**else if(counter == 30)**

**{**

**counter = counter + 1;**

**}**

**else if(counter == 40)**

**{**

**setLocation(getX(),getY() + interval);**

**counter = counter + 1;**

**}**

**else if(counter == 50)**

**{**

**setLocation(getX(),getY() + interval);**

**counter = counter + 1;**

**}**

**else if(counter == 60)**

**{**

**setLocation(getX(),getY() + interval);**

**counter = counter + 1;**

**}**

**else if(counter == 70)**

**{**

**counter = counter + 1;**

**}**

**else if(counter == 80)**

**{**

**counter = 0;**

**}**

**else {**

**counter = counter + 1;**

**}**

**}**

**}**

**Cookie**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class Cookie here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Cookie extends Actor

{

/\*\*

\* Act - do whatever the Cookie wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

int counter = 0;

int interval = 2;

public void act()

{

if(counter == 0)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 10)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 20)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 30)

{

counter = counter + 1;

}

else if(counter == 40)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 50)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 60)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 70)

{

counter = counter + 1;

}

else if(counter == 80)

{

counter = 0;

}

else {

counter = counter + 1;

}

}

}

**Counter**

**import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)**

**/\*\***

**\* A Counter class that allows you to display a numerical value on screen.**

**\***

**\* The Counter is an actor, so you will need to create it, and then add it to**

**\* the world in Greenfoot. If you keep a reference to the Counter then you**

**\* can adjust its value. Here's an example of a world class that**

**\* displays a counter with the number of act cycles that have occurred:**

**\***

**\* <pre>**

**\* class CountingWorld**

**\* {**

**\* private Counter actCounter;**

**\***

**\* public CountingWorld()**

**\* {**

**\* super(600, 400, 1);**

**\* actCounter = new Counter("Act Cycles: ");**

**\* addObject(actCounter, 100, 100);**

**\* }**

**\***

**\* public void act()**

**\* {**

**\* actCounter.setValue(actCounter.getValue() + 1);**

**\* }**

**\* }**

**\* </pre>**

**\***

**\* @author Neil Brown and Michael KÃ¶lling**

**\* @version 1.0**

**\*/**

**public class Counter extends Actor**

**{**

**private static final Color transparent = new Color(0,0,0,0);**

**private GreenfootImage background;**

**private int value;**

**private int target;**

**private String prefix;**

**public Counter()**

**{**

**this(new String());**

**}**

**/\*\***

**\* Create a new counter, initialised to 0.**

**\*/**

**public Counter(String prefix)**

**{**

**background = getImage(); // get image from class**

**value = 0;**

**target = 0;**

**this.prefix = prefix;**

**updateImage();**

**}**

**/\*\***

**\* Animate the display to count up (or down) to the current target value.**

**\*/**

**public void act()**

**{**

**if (value < target) {**

**value++;**

**updateImage();**

**}**

**else if (value > target) {**

**value--;**

**updateImage();**

**}**

**}**

**/\*\***

**\* Add a new score to the current counter value. This will animate**

**\* the counter over consecutive frames until it reaches the new value.**

**\*/**

**public void add(int score)**

**{**

**target += score;**

**}**

**/\*\***

**\* Return the current counter value.**

**\*/**

**public int getValue()**

**{**

**return target;**

**}**

**/\*\***

**\* Set a new counter value. This will not animate the counter.**

**\*/**

**public void setValue(int newValue)**

**{**

**target = newValue;**

**value = newValue;**

**updateImage();**

**}**

**/\*\***

**\* Sets a text prefix that should be displayed before**

**\* the counter value (e.g. "Score: ").**

**\*/**

**public void setPrefix(String prefix)**

**{**

**this.prefix = prefix;**

**updateImage();**

**}**

**/\*\***

**\* Update the image on screen to show the current value.**

**\*/**

**private void updateImage()**

**{**

**GreenfootImage image = new GreenfootImage(background);**

**GreenfootImage text = new GreenfootImage(prefix + value, 22, Color.BLACK, transparent);**

**if (text.getWidth() > image.getWidth() - 20)**

**{**

**image.scale(text.getWidth() + 20, image.getHeight());**

**}**

**image.drawImage(text, (image.getWidth()-text.getWidth())/2,**

**(image.getHeight()-text.getHeight())/2);**

**setImage(image);**

**}**

**}**

**CounterP**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* A Counter class that allows you to display a numerical value on screen.

\*

\* The Counter is an actor, so you will need to create it, and then add it to

\* the world in Greenfoot. If you keep a reference to the Counter then you

\* can adjust its value. Here's an example of a world class that

\* displays a counter with the number of act cycles that have occurred:

\*

\* <pre>

\* class CountingWorld

\* {

\* private Counter actCounter;

\*

\* public CountingWorld()

\* {

\* super(600, 400, 1);

\* actCounter = new Counter("Act Cycles: ");

\* addObject(actCounter, 100, 100);

\* }

\*

\* public void act()

\* {

\* actCounter.setValue(actCounter.getValue() + 1);

\* }

\* }

\* </pre>

\*

\* @author Neil Brown and Michael KÃ¶lling

\* @version 1.0

\*/

public class CounterP extends Actor

{

private static final Color transparent = new Color(0,0,0,0);

private GreenfootImage background;

private int value;

private int target;

private String prefix;

public CounterP()

{

this(new String());

}

/\*\*

\* Create a new counter, initialised to 0.

\*/

public CounterP(String prefix)

{

background = getImage(); // get image from class

value = 0;

target = 0;

this.prefix = prefix;

updateImage();

}

/\*\*

\* Animate the display to count up (or down) to the current target value.

\*/

public void act()

{

if (value < target) {

value++;

updateImage();

}

else if (value > target) {

value--;

updateImage();

}

}

/\*\*

\* Add a new score to the current counter value. This will animate

\* the counter over consecutive frames until it reaches the new value.

\*/

public void add(int score)

{

target += score;

}

/\*\*

\* Return the current counter value.

\*/

public int getValue()

{

return target;

}

/\*\*

\* Set a new counter value. This will not animate the counter.

\*/

public void setValue(int newValue)

{

target = newValue;

value = newValue;

updateImage();

}

/\*\*

\* Sets a text prefix that should be displayed before

\* the counter value (e.g. "Score: ").

\*/

public void setPrefix(String prefix)

{

this.prefix = prefix;

updateImage();

}

/\*\*

\* Update the image on screen to show the current value.

\*/

private void updateImage()

{

GreenfootImage image = new GreenfootImage(background);

GreenfootImage text = new GreenfootImage(prefix + value, 22, Color.BLACK, transparent);

if (text.getWidth() > image.getWidth() - 20)

{

image.scale(text.getWidth() + 20, image.getHeight());

}

image.drawImage(text, (image.getWidth()-text.getWidth())/2,

(image.getHeight()-text.getHeight())/2);

setImage(image);

}

}

**Enemy**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class Enemy here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Enemy extends Actor

{

private int speed;

private int leftTurn;

private int rightTurn;

private int up;

private int down;

public GreenfootImage image1 = new GreenfootImage("enemy-left-removebg-preview.png");

public GreenfootImage image2 = new GreenfootImage("enemy-right.png");

public Enemy(){

this(0,0,0,0,0);

}

public Enemy(

int leftTurnChanged,

int rightTurnChanged,

int speedChanged,

int upChanged,

int downChanged)

{

setLeftTurn(leftTurnChanged);

setRightTurn(rightTurnChanged);

setSpeed(speedChanged);

setUp(upChanged);

setDown(downChanged);

GreenfootImage image = getImage();

image.scale(image.getWidth()/2, image.getHeight()/2);

image1.scale(image1.getWidth()/2, image1.getHeight()/2);

image2.scale(image2.getWidth()/2, image2.getHeight()/2);

if(leftTurnChanged == 0) ChangeImage();

}

public void act()

{

if(leftTurn != 0 || rightTurn != 0)

{

setLocation(getX() + speed,getY());

Actor actor = getOneIntersectingObject(null);

if(actor!=null)

{

actor.setLocation(actor.getX()+speed,actor.getY());

}

if(atTurningPointX())

{

speed = -speed;

ChangeImage();

}

}

if(up != 0 || down != 0)

{

setLocation(getX(),getY()+ speed);

Actor actor = getOneIntersectingObject(null);

if(actor!=null)

{

actor.setLocation(actor.getX(),actor.getY()+speed);

}

if(atTurningPointY())

{

speed = -speed;

}

}

}

public void ChangeImage()

{

if(getImage().equals(image1))

{

setImage(image2);

}

else

{

setImage(image1);

}

}

public void setLeftTurn(int leftTurnChanged)

{

leftTurn = leftTurnChanged;

}

public void setRightTurn(int rightTurnChanged)

{

rightTurn = rightTurnChanged;

}

public void setSpeed(int speedChanged)

{

speed = speedChanged;

}

public void setUp(int upChanged)

{

up = upChanged;

}

public void setDown(int downChanged)

{

down = downChanged;

}

public boolean atTurningPointX()

{

return (getX() <= leftTurn || getX() >= rightTurn);

}

public boolean atTurningPointY()

{

return (getY() <= up || getY() >= down);

}

}

**Ghost**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class TestCode here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Ghost extends Actor

{

private final int GRAVITY = 1;

private final int STEP = 3;

private final int ABOVEOFFSET = 20;

private int velocity;

private int jumpStrength = 15;

private boolean hasSword = false;

public Ghost()

{

velocity = 0;

GreenfootImage image = getImage();

image.scale(image.getWidth()/2, image.getHeight()/2);

}

public void act()

{

fall();

if(Greenfoot.isKeyDown("w") && isOnSolidGround()) jump();

move();

PlayWorld();

}

public void PlayWorld()

{

if(this.getWorld() instanceof Level\_One)

{

Level\_One myWorld = (Level\_One) getWorld();

if(CheckPortal())LevelTwo();

if(CheckEnemy())

{

DeathScreenLevelOne();

}

if(CheckJewel())

{

myWorld.IncrementJewel();

}

if(CheckClock())

{

myWorld.IncrementClock();

}

}

else if (this.getWorld() instanceof Level\_Two)

{

if(CheckPortal())WinLevel();

Level\_Two myWorld = (Level\_Two) getWorld();

if(CheckEnemy())

{

DeathScreenLevelTwo();

}

if(CheckJewel())

{

myWorld.IncrementJewel();

}

if(CheckClock())

{

myWorld.IncrementClock();

}

}

if(CheckSword()) hasSword = true;

if(CheckPotion())SetJump();

if(CheckCookie())Win();

}

public void fall()

{

setLocation(getX(),getY() + velocity);

if(isOnSolidGround())

{

velocity = 0;

while(isOnSolidGround())

{

setLocation(getX(),getY() -1);

}

setLocation(getX(),getY()+1);

}

else if (velocity < 0 && didBumpHead())

{

velocity = 0;

while(didBumpHead())

{

setLocation(getX(),getY() + 1);

}

}

else velocity += GRAVITY;

}

public void jump()

{

velocity = -jumpStrength;

Greenfoot.playSound("jump.mp3");

}

//Checks if certain keys are pressed and moves the actor

//Unless the actor is touching another object stopping it from moving in that direction

//Also changes the image of the actor to the corresponding side

public void move()

{

int y = getY();

int x = getX();

if(Greenfoot.isKeyDown("a") && canMoveLeft())

{

x -=STEP;

if(hasSword)

{

setImage("blob\_sword\_left.png");

}

else

{

setImage("blob\_left-removebg-preview.png");

}

GreenfootImage image = getImage();

image.scale(image.getWidth()/2, image.getHeight()/2);

}

if(Greenfoot.isKeyDown("d")&& canMoveRight())

{

x +=STEP;

if(hasSword)

{

setImage("blob\_sword\_right.png");

}

else

{

setImage("blob\_right-removebg-preview.png");

}

GreenfootImage image = getImage();

image.scale(image.getWidth()/2, image.getHeight()/2);

}

setLocation(x,y);

}

//Checks if the actor is standing on an object

//If it is, then it stop the gravity

public boolean isOnSolidGround()

{

boolean isOnGround = false;

if(getY() > getWorld().getHeight() - ABOVEOFFSET) isOnGround = true;

int imageWidth = getImage().getWidth();

int imageHeight = getImage().getHeight();

if(getOneObjectAtOffset(imageWidth/-2,imageHeight/2,BetterPlatform.class) != null ||

getOneObjectAtOffset(imageWidth/2,imageHeight/2,BetterPlatform.class) != null)

isOnGround = true;

return isOnGround;

}

//Checks if actor's head touched an object stopping it from entering the object

public boolean didBumpHead()

{

boolean bumpedHead = false;

int imageWidth = getImage().getWidth();

int imageHeight = getImage().getHeight();

if(getOneObjectAtOffset(imageWidth/-2,imageHeight/-2,BetterPlatform.class) != null ||

getOneObjectAtOffset(imageWidth/2,imageHeight/-2,BetterPlatform.class) != null)

bumpedHead = true;

return bumpedHead;

}

public boolean canMoveLeft()

{

boolean canMoveLeft = true;

int imageWidth = getImage().getWidth();

int imageHeight = getImage().getHeight();

if(getOneObjectAtOffset(imageWidth/-2 - STEP,imageHeight/-2,BetterPlatform.class) != null ||

getOneObjectAtOffset(imageWidth/-2 - STEP,imageHeight/2 - 1,BetterPlatform.class) != null)

canMoveLeft = false;

return canMoveLeft;

}

public boolean canMoveRight()

{

boolean canMoveRight = true;

int imageWidth = getImage().getWidth();

int imageHeight = getImage().getHeight();

if(getOneObjectAtOffset(imageWidth/2 + STEP,imageHeight/-2,BetterPlatform.class) != null ||

getOneObjectAtOffset(imageWidth/2 + STEP,imageHeight/2 - 1,BetterPlatform.class) != null)

canMoveRight = false;

return canMoveRight;

}

//Moves the actor to the death screen fro level one

public void DeathScreenLevelOne()

{

Level\_One myWorld = (Level\_One) getWorld();

myWorld.SetStageStart(0);

GameOverScreen gameOver = new GameOverScreen();

Greenfoot.setWorld(gameOver);

}

//Moves the actor to the death screen fro level two

public void DeathScreenLevelTwo()

{

Level\_Two myWorld = (Level\_Two) getWorld();

myWorld.SetStageStart(0);

GameOverScreen gameOver = new GameOverScreen();

Greenfoot.setWorld(gameOver);

}

//Moves the actor to level two

public void LevelTwo()

{

Level\_One myWorld = (Level\_One) getWorld();

myWorld.SetStageStart(0);

Level\_Two level\_Two = new Level\_Two();

Greenfoot.setWorld(level\_Two);

}

//Moves the actor to the final level

public void WinLevel()

{

Level\_Two myWorld = (Level\_Two) getWorld();

myWorld.SetStageStart(0);

WinLevel winLevel = new WinLevel();

Greenfoot.setWorld(winLevel);

}

public void Win()

{

WinLevel myWorld = (WinLevel) getWorld();

myWorld.SetStageStart(0);

YouWonScreen win = new YouWonScreen();

Greenfoot.setWorld(win);

}

//Checks if touching portal

public boolean CheckPortal()

{

boolean touchingPortal = false;

if(getOneIntersectingObject(PortalToNextLvl.class) != null)

{

touchingPortal = true;

Greenfoot.playSound("portal.wav");

}

return touchingPortal;

}

//Checks if touching jewel and removes the jewel

public boolean CheckJewel()

{

boolean touchingJewel = false;

Actor jewel = getOneIntersectingObject(Jewel.class);

if(jewel != null)

{

getWorld().removeObject(jewel);

touchingJewel = true;

Greenfoot.playSound("pickup.wav");

}

return touchingJewel;

}

//Checks if touching Clock and removes the clock

public boolean CheckClock()

{

boolean touchingClock = false;

Actor clock = getOneIntersectingObject(Clock.class);

if(clock != null)

{

getWorld().removeObject(clock);

touchingClock = true;

Greenfoot.playSound("clockgrab.wav");

}

return touchingClock;

}

//Checks if touching enemy, if actor has a sword then the enemy is removed

public boolean CheckEnemy()

{

boolean touchingEnemy = false;

Actor enemy = getOneIntersectingObject(Enemy.class);

if(enemy != null && !hasSword)

{

Greenfoot.playSound("enemyhityou.wav");

touchingEnemy = true;

}

else if ( enemy != null && hasSword)

{

Greenfoot.playSound("enemygettinghit.wav");

getWorld().removeObject(enemy);

}

return touchingEnemy;

}

//Checks if touching sword and removes the sword

public boolean CheckSword()

{

boolean touchingSword = false;

Actor sword = getOneIntersectingObject(Sword.class);

if(sword != null)

{

getWorld().removeObject(sword);

//Greenfoot.playSound("enemyhityou.wav");

touchingSword = true;

}

return touchingSword;

}

//Allows for changing of the jumpStrength when potion is picked up

public void SetJump()

{

jumpStrength = jumpStrength + 10;

}

//Checks if touching potion and removes the potion

public boolean CheckPotion()

{

boolean touchingPotion = false;

Actor potion = getOneIntersectingObject(Potion.class);

if(potion != null)

{

getWorld().removeObject(potion);

touchingPotion = true;

Greenfoot.playSound("clockgrab.wav");

}

return touchingPotion;

}

//Checks if touching cookie and removes the cookie

public boolean CheckCookie()

{

boolean touchingCookie = false;

Actor cookie = getOneIntersectingObject(Cookie.class);

if(cookie != null)

{

getWorld().removeObject(cookie);

touchingCookie = true;

Greenfoot.playSound("biscuitgrab.wav");

}

return touchingCookie;

}

}

**JEWELCOUNTER**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class JEWELCOUNTER here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class JEWELCOUNTER extends Actor

{

/\*\*

\* Act - do whatever the JEWELCOUNTER wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

public void act()

{

// Add your action code here.

}

}

**Jewel**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class Jewel here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Jewel extends Actor

{

/\*\*

\* Act - do whatever the Jewel wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

int counter = 0;

int interval = 2;

public Jewel()

{

GreenfootImage image = getImage();

image.scale(image.getWidth()/2, image.getHeight()/2);

}

public void act()

{

if(counter == 0)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 10)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 20)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 30)

{

counter = counter + 1;

}

else if(counter == 40)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 50)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 60)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 70)

{

counter = counter + 1;

}

else if(counter == 80)

{

counter = 0;

}

else {

counter = counter + 1;

}

}

}

**LivesHearts**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class LivesHearts here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class LivesHearts extends Actor

{

/\*\*

\* Act - do whatever the LivesHearts wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

public void act()

{

// Add your action code here.

}

}

**PlayAgain**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class PlayAgain here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class PlayAgain extends Actor

{

/\*\*

\* Act - do whatever the PlayAgain wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

public PlayAgain()

{

GreenfootImage image = getImage();

image.scale(image.getWidth(), image.getHeight());

}

public void act()

{

if (Greenfoot.mousePressed(this))

{

Greenfoot.setWorld(new Level\_One());

}

}

}

**PortalToNextLvl**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class PortalToNextLvl here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class PortalToNextLvl extends Actor

{

public PortalToNextLvl()

{

GreenfootImage image = getImage();

image.scale(image.getWidth()/2, image.getHeight()/2);

}

public void act()

{

// Add your action code here.

}

}

**Potion**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class Potion here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Potion extends Actor

{

/\*\*

\* Act - do whatever the Potion wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

int counter = 0;

int interval = 2;

public void act()

{

if(counter == 0)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 10)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 20)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 30)

{

counter = counter + 1;

}

else if(counter == 40)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 50)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 60)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 70)

{

counter = counter + 1;

}

else if(counter == 80)

{

counter = 0;

}

else {

counter = counter + 1;

}

}

}

**Sword**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class Sword here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class Sword extends Actor

{

/\*\*

\* Act - do whatever the Sword wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

int counter = 0;

int interval = 2;

public Sword()

{

GreenfootImage image = getImage();

image.scale(image.getWidth()/2, image.getHeight()/2);

}

public void act()

{

if(counter == 0)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 10)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 20)

{

setLocation(getX(),getY() - interval);

counter = counter + 1;

}

else if(counter == 30)

{

counter = counter + 1;

}

else if(counter == 40)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 50)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 60)

{

setLocation(getX(),getY() + interval);

counter = counter + 1;

}

else if(counter == 70)

{

counter = counter + 1;

}

else if(counter == 80)

{

counter = 0;

}

else {

counter = counter + 1;

}

}

}

**TIMEDISPLAY**

import greenfoot.\*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/\*\*

\* Write a description of class TIMEDISPLAY here.

\*

\* @author (your name)

\* @version (a version number or a date)

\*/

public class TIMEDISPLAY extends Actor

{

/\*\*

\* Act - do whatever the TIMEDISPLAY wants to do. This method is called whenever

\* the 'Act' or 'Run' button gets pressed in the environment.

\*/

public void act()

{

// Add your action code here.

}

}

**SimpleTimer**

/\*\*

\* A simple timer class that allows you to keep track of how much time

\* has passed between events.

\*

\* You use this class by creating a timer as a member field in your actor (or whatever):

\* <pre>

\*

\* private SimpleTimer timer = new SimpleTimer();

\* </pre>

\*

\* Then when you want to start the timer (for example, when a shot is fired), you call the mark() method:

\*

\* <pre>

\*

\* timer.mark();

\* </pre>

\*

\* Thereafter, you can use the millisElapsed() method to find out how long it's been since mark()

\* was called (in milliseconds, i.e. thousandths of a second). So if you want to only allow the player to fire a shot every second, you

\* could write:

\*

\* <pre>

\*

\* if (timer.millisElapsed() > 1000 && Greenfoot.isKeyDown("space"))

\* {

\* // Code here for firing a new shot

\* timer.mark(); // Reset the timer

\* }

\* </pre>

\*

\* @author Neil Brown

\* @version 1.0

\*/

public class SimpleTimer

{

private long lastMark = System.currentTimeMillis();

/\*\*

\* Marks the current time. You can then in future call

\* millisElapsed() to find out the elapsed milliseconds

\* since this mark() call was made.

\*

\* A second mark() call will reset the mark, and millisElapsed()

\* will start increasing from zero again.

\*/

public void mark()

{

lastMark = System.currentTimeMillis();

}

/\*\*

\* Returns the amount of milliseconds that have elapsed since mark()

\* was last called. This timer runs irrespective of Greenfoot's

\* act() cycle, so if you call it many times during the same Greenfoot frame,

\* you may well get different answers.

\*/

public int millisElapsed()

{

return (int) (System.currentTimeMillis() - lastMark);

}

}